Name:	
ID:	

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Quiz 10 Solutions **Spring 2020, CU-Boulder**

Standard 20. Consider the weighed interval scheduling problem for the job list

$$A = [(0,3;3),(2,4;2),(3,5;1),(3,6;3),(2,7;5)]$$

of (start time, finish time; value) triples. Fill in the values of the following DP table, where $OPT(i)$ is the value of optimal solution to the problem consisting of the first i jobs.

triple	$OPT(i)$
(0, 3; 3)	3
(2, 4; 2)	3
(3, 5; 1)	4
(3, 6; 3)	6
(2, 7; 5)	6

OPT(1): base case, so we take job 1, which has weight 3.

OPT(2): Taking job 2 results in a weight of 2, and excludes job 1, while leaving job 2 allows us to take OPT(1)=3, which we do. So OPT(2)=3.

OPT(3): Taking job 3 conflicts with job 2, but not with any prior job, so we get $1 + OPT(1) = 4$. Leaving job 3 leaves us with $OPT(2) = 3 < 4$, so we take job 3 and $OPT(3) = 4$.

OPT(4): Taking job 4 conflicts with the prior jobs until job 1, resulting in a value of $3 + 3 = 6$. Leaving job 4 leaves us with $OPT(3) = 4 < 6$, so we take job 4 and $OPT(4) = 6$.

OPT(5): Taking job 5 conflicts with all other jobs resulting in a value of 5. Leaving job 5 result in $OPT(4) = 6 > 5$, so we leave job 5.

Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Quiz 10 Solutions **Spring 2020, CU-Boulder**

Standard 21. Recall the sequence alignment problem where the cost of *sub* and the cost of *indel* are all 1. Given the following table of optimal cost of aligning the strings EXPONEN and POLYNO, draw the backward path consisting of backward edges to find the minimal-cost set of edit operations that transforms EXPONEN to POLYNO. Besides indicating the backward path, you must also give the minimal-cost set of edit operations.

	$-$	P	O	L	Y	N	O
$-$	0	1	2	3	4	5	6
E	↑1	1	2	3	4	5	6
X	↑2	2	2	3	4	5	6
P	3	↖2	3	3	4	5	6
O	4	3	↖2	3	4	5	5
N	5	4	3	↖3	4	4	5
E	6	5	4	4	↖4	5	5
N	7	6	5	5	5	↖4	←5

The optimal alignment is

```
EXPONEN-
--POLYNO
```

And the optimal set of edit operations to get from EXPONEN to POLYNO is: del(E), del(X), noop(P), noop(O), sub(N→L), sub(E→Y), noop(N), ins(O).

Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Quiz 10 Solutions **Spring 2020, CU-Boulder**

Standard 22. Suppose you have 8 stairs to climb. You may choose to jump up either $1, 2$, or 3 stairs. Your starting position is on the ground floor and not on the first stair. Your goal is to count the number of ways to climb the stairs.

Use dynamic programming to fill in a table that counts number of ways to climb from the ground floor to each stair j for $1 \leq j \leq 8$.

Let $C(i)$ denote the number of ways to climb from stair 1 to stair i. Then by considering the last jump, we get the recurrence

$$C(i) = C(i-1) + C(i-2) + C(i-3)$$

where we define $C(i) = 0$ for $i < 0$ and $C(0) = 1$. These conventions are both convenient to get correct counts, and they make sense: there's one way to climb no stairs, namely don't climb; and there are no ways to climb a negative number of stairs, since they don't exist! (An alternative is to only define $C(i)$ for $i \geq 1$, and to have three base cases for $i = 1, 2, 3$. There are also other alternatives that will work to give the same correct counts.)

Then we have the table

i	0	1	2	3	4	5	6	7	8
$C(i)$	1	1	2	4	7	13	24	44	81

Hand check for the first four:

1. $C(1)$: there is only 1 way to climb 1 stair, namely a jump of size 1
2. $C(2)$: there are two ways, namely $1 + 1$ or 2
3. $C(3)$: There are four ways, namely $1 + 1 + 1$ or $1 + 2$ or $2 + 1$ or 3
4. $C(4)$: There are seven ways, namely $1+1+1+1$, $2+1+1$, $1+2+1$, $1+1+2$, $2+2$, $1+3$, $3+1$.